# MACHINE LEARNING PROJECT

- Fake Review Detection


✓ Prepared by:
Sagar Lamichhane

## Problem Definition:

Given the review of three Indian Restaurants, the task is to predict whether the review is Fake or Real about the Restaurants. This is a typical supervised learning task where given a text string, we have to categorize the text string into predefined categories.

## Solution

To solve this problem, we will follow the typical machine learning pipeline. We will first import the required libraries and the dataset. We will then do exploratory data analysis to see if we can find any trends in the dataset. Next, we will perform text preprocessing to convert textual data to numeric data that can be used by a machine learning algorithm. Finally, we will use machine learning algorithms to train and test our sentiment analysis models.

## Importing the Required Libraries

The first step as always is to import the required libraries:

```
import sys
import nltk
import sklearn
import pandas
import numpy
```

## Importing the Dataset

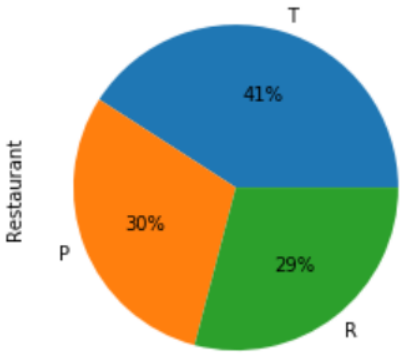
To import the excel dataset, I have used Pandas read_excel function. On using head() and info() method, the dataset outputs the general information as below:

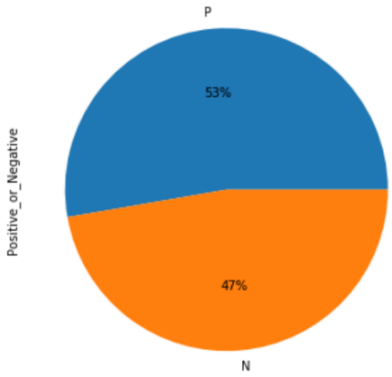
```
RangeIndex: 110 entries, 0 to 109
Data columns (total 5 columns):
Email                    110 non-null object
Restaurant               110 non-null object
Real_or_Fake             110 non-null object
Positive_or_Negative     110 non-null object
Reviews                  110 non-null object
dtypes: object(5)
memory usage: 4.4+ KB
None
                     Email Restaurant Real_or_Fake Positive_or_Negative  \
0  sagar.lamichhane@ttu.edu          T            F                     P
1  sagar.lamichhane@ttu.edu          T            F                     N
2  sagar.lamichhane@ttu.edu          T            F                     N
3  sagar.lamichhane@ttu.edu          T            F                     N
4  sagar.lamichhane@ttu.edu          T            R                     P

                                          Reviews
0  The place is really great. Owner is from my vi...
1  For me, it was the worst place to have an expe...
2  It was a nightmare to go there and spend my ti...
3  I wonder why such place still exists. Why woul...
4  The food is very good and a break from the usu...
```

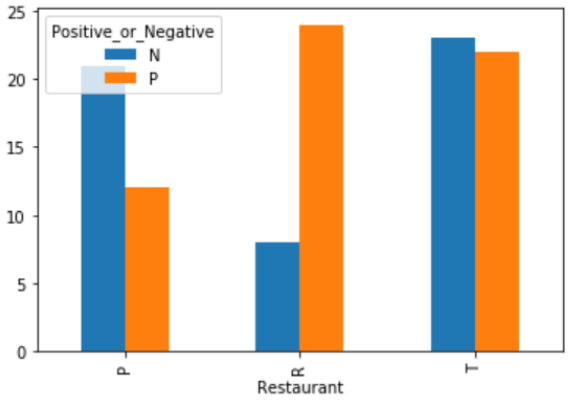
## Data Analysis

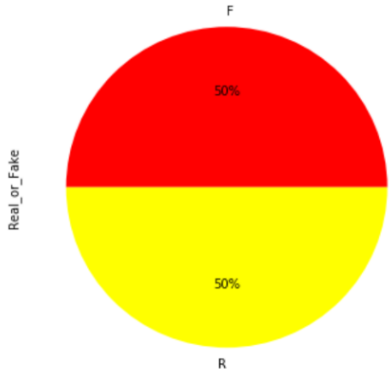
Let's explore the dataset a bit to see if we can find any trends. But before that, I will change the default plot size to have a better view of the plots.

First, let's find out the distribution of our data based upon the total number of reviews of the restaurants in percentages. Next, let's find out the total number of positive and negative reviews on the given data. Then, lets plot a bar based upon Positive and Negative reviews of each restaurants. Finally, lets calculate the total number of fake and real reviews.




Here, Restaurant T represents Tikka shack, R represents Royal Indian, and P represents India Palace.

From the above graphs, it is found out that our data contains highest number of reviews of Tikka Shack (41%), followed by India Palace (30%) and least number of reviews of Royal Indian (21%).

Also, our data contains almost equal amount with slight more of positive than negative reviews.

Surprisingly, our data has equal amount of both Real and Fake reviews containing each of 55.

## Preprocess and Data Cleaning

Reviews contain many slang words and punctuation marks. We need to clean them before they can be used for training the machine learning model. However, before cleaning the review, let's assign binary values to our class levels; 0 for Fake and 1 for Real. It is then printed to make sure the values are assigned correctly. Also, the first 10 reviews are printed.

Next step is to remove unnecessary information out of the datasets. Following unwanted materials are removed:

a. Information containing words such as email address, URLs, phone numbers, other numbers, and symbols using regular expressions library.
b. Punctuations as well as leading and trailing whitespaces are removed.
c. All words are then converted into lower case.
d. Stopwords (most occurring words that doesn't provide any valuable information) such as a, the, he, she, some, etc. are removed.
e. Finally, stems words (words containing ing, tenses, etc.) are removed by using porter stemmer.

Updated reviews are then printed where it can be noticed that unnecessary information is removed and filtered out.

## Generating Features

Feature engineering is the process of using domain knowledge of the data to create features for machine learning algorithms. In this project, the words in each Reviews will be our features. For this purpose, it will be necessary to tokenize each word which follows the **bag-of-words** model. Then the 100 most common words are used as features using nltk libraries. Then the total number of words and 15 most common words are printed.

Since the given dataset is small, I am using the 100 most common words as the feature.

A function named find_features is created that determines which of the 100-word features are contained in the review. It is done by passing message as a parameter and tokenizing word. As an example, first index of the dataset is used and on printing we can see the updated information.

Then, the process is followed to find features in all listed reviews. Zip function is used to map the similar index of multiple containers. Seed is defined for reproducibility by setting the random seed through NumPy. The reviews are now shuffled. The function find_features() is called for each review containing text and labels.

Now, the featuresets are split into training and testing datasets using sklearn. We split our data on 75% for Training and 25% for Testing purposes.

## Scikit-Learn Classifiers with NLTK

Now that we have our dataset, we can start building algorithms! Let's start with a simple linear support vector classifier, then expand to other algorithms. We'll need to import each algorithm we plan on using from sklearn. We also need to import some performance metrics, such as accuracy_score and classification_report.

First, let's import all the available **sk-learn** classifiers such as **KNeighborsClassifier, DecisionTreeClassifier, RandomForestClassifier, LogisticRegression, SGDClassifier, MultinomialNB, SVC, and classification_report, accuracy_score, confusion_matrix** to check which one produces better accuracy on our algorithm.

To define the models to train, we have included the lists of names and classifiers of all available **sk-learn** classifiers. Then the classifiers are zipped on a dictionary of models of names and classifiers. The name of classifiers along with the percentages of their accuracy are then printed.

```
K Nearest Neighbors Accuracy: 60.71428571428571
Decision Tree Accuracy: 57.14285714285714
Random Forest Accuracy: 60.71428571428571
Logistic Regression Accuracy: 53.57142857142857
SGD Classifier Accuracy: 67.85714285714286
Naive Bayes Accuracy: 57.14285714285714
SVM Linear Accuracy: 67.85714285714286
```

On comparing the accuracy of the classifiers, it is found that Logistic Regression has the lowest accuracy percentage with only 53.57, whereas SGD, and SVM classifiers have same amount of highest percentage.

### Ensemble methods - Voting classifier

In this method, there will be a hard voting where all the algorithms are combined and ask them to vote whether they think the particular review is classified as Real or Fake. Since we have 7 classifiers, if any review gets 4 or more vote as Real or Fake, then it is classified accordingly. This process is similar with the one we have used to get the accuracy of the seven classifiers.

```
Voting Classifier: Accuracy: 67.85714285714286
```

On hard voting, the accuracy percentage was found to be 67.86.

### Class label Prediction

In this method, we will check where our algorithm went right and where we got wrong. To do so, lets unzip the features and labels from testing and generate the predictions using nltk_ensamble method on calling classify_many by passing txt_features.

## Confusion Matrix and Classification Report

Finally, to evaluate the performance of the machine learning models, we can use classification metrics such as confusion matrix, F1 measure, accuracy, etc.

To find the values for these metrics, we can use classification_report, confusion_matrix, and accuracy_score utilities from the sklearn.metrics library.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.77 | 0.59 | 0.67 | 17 |
| 1 | 0.53 | 0.73 | 0.62 | 11 |
| accuracy | | | 0.64 | 28 |
| macro avg | 0.65 | 0.66 | 0.64 | 28 |
| weighted avg | 0.68 | 0.64 | 0.65 | 28 |

| | | predicted | |
|---|---|---|---|
| | | **Real** | **Fake** |
| actual | Real | 10 | 7 |
| | Fake | 3 | 8 |

The classification report shows the precision, recall, f1-score and support. It reveals the false positive and false negative scores of both Real (1) and Fake (0) classes.

The confusion matrix is containing the labels and predictions of both Real and Predicted data. From the figure, it is found that 10 times Real predicted was real and 3 times predicted was Fake. Likewise, on prediction of Fake, 7 times predicted was Real and 8 times prediction was Fake.

**Note:** Since our data is small, the percentage of accuracy will be lower and keeps on fluctuating as compared with the big data.